Claims

1. (currently amended) An object execution system supporting passing of an object reference in a method invocation delivered via a queued message, the system comprising:

an object configuration store containing object properties information representing properties of at least first and second object classes executable in the system, the object properties information designating the first and second objects as supporting queued method invocation, the second object class having a method with a parameter for passing an object reference;

a method invocation recording facility operative responsive to request of a client program to supply method invocations recorders for object instances of object classes designated as supporting queued method invocations;

in a first transaction, a first method invocations recorder supplied by the method invocation recording facility at request of the client program for the first object class; and

in the first transaction, a second method invocations recorder supplied by the method invocation recording facility at request of the client program for the second object class, the second method invocations recorder operating in response to a method invocation in which an object reference to the first method invocations recorder is passed to cause a data stream representation of the first method invocations recorder to be marshaled into a method invocations message for submission into a message queue associated with the second object class;

a method invocation play-back facility operative responsive to a queued method invocations message to supply method invocation players for object instances of object classes designated as supporting queued method invocations;

in a second transaction, a first method invocation player supplied by the method invocation play-back facility in response to the method invocations message queued to the message queue associated with the second object class, the first method invocations player operating in response to the method invocations message to unmarshal the date stream representation and create therefrom a copy of the first method invocations recorder, and to pass an object reference to the copy of the first method invocations recorder as a parameter of a method invocation to an object of the second object class;

wherein an object reference to the copy of the first method invocation recorder is passed as a parameter of the method invocation to an object of the second object class so an object of the second object class can record results for an object of the first object class.

2. (original) The object execution system of claim 1 wherein the first method invocations recorder is marshaled into the data stream representation via a marshal-by-value operation, such that the copy of the first method invocations recorder can be created on a separate computing machine.

3. (original) The object execution system of claim 1 further comprising:

a persistence interface associated with the first method invocations recorder operating when invoked to persistently write the data stream representation of the first method invocations recorder; and

the second method invocations recorder invoking the persistence interface to cause marshaling of the first method invocations recorder into the data stream representation.

4. (original) The object execution system of claim 1 wherein the second method invocations recorder further operates to cause an identification of a message queue associated with the first object class to be marshaled into the data stream representation.

5. (original) The object execution system of claim 1 further supporting execution of distributed objects across remote machines in a distributed computing system.

6. (currently amended) A computer-implemented method of yielding transaction processing based results from processing work of a first queued component to a second queued component, where the work of the first queued component is initiated by method invocations delivered via a first message queue, and the second queued component is dispatched method invocations delivered into a second message queue, the method comprising:

in a first transaction responsive to a client program issuing a first set of method invocations for the first queued component, marshaling data for the method invocations of the first set into a message to be enqueued into the first message queue; and when marshaling an interface pointer reference to the second queued component in any of the method invocations issued by the client program for the first queued component, incorporating interface passing information in the data marshaled into the message, the interface passing information designating to enqueue any method invocation by the first queued component on an interface of the second queued component referenced by the interface pointer reference into the second message queue during a second transaction.

7. (original) The method of claim 6 further comprising, responsive to the first queued component issuing a second set of method invocations on the interface of the second queued component referenced by the interface pointer reference, enqueueing the method invocations of the second set into the second message queue.

8. (original) The method of claim 6 further comprising:

passing the interface pointer reference in queued method invocations to multiple further queued components; and

responsive to the first queued component and the multiple queued components issuing sets of method invocations on the interface of the second queued component, enqueueing the method invocations of each such set into the second message queue.

9. (currently amended) A computer-implemented transaction processing method of yielding results from processing work of a queued component to a persist-able object, where the work of the queued component is initiated by method invocations delivered via a first message queue, the method comprising:

in a first transaction,

responsive to a client program issuing a set of method invocations for the queued component, marshaling data for the method invocations of the set into a message to be enqueued into the first message queue~~;~~, and

when marshaling a reference for calling methods on the persist-able object in any of the method invocations issued by the client program for the queued component, persisting the persist-able object into an object-representative data stream and incorporating the object-representative data stream in the data marshaled into the message~~;~~, submitting the message to the first message queue; and

in a second transaction, ~~at a later time of~~ processing the message from the first message queue, unmarshaling the data for the method invocations from the message, re-creating the persist-able object from the object-representative data stream, issuing the set of method invocations to the queued component, and passing a reference for calling methods on the re-created persist-able object in said any method invocation to the queued component.

10. (original) The method of claim 9 wherein the persist-able object is a method invocations recorder of a second queued component.

11. (original) The method of claim 9 wherein the references for calling methods are interface pointers referencing an interface exposed by the persist-able object.

12. (currently amended) In a distributed computing system having a multiplicity of client machines and a server machine, a method of conveying results of processing queued method invocations by a server-side queued component of a component-based server program on the server machine to a client-specific component-based program, the method comprising:

associating a first message queue with the server-side queued component and a second message queue with a client-specific queued component of the client-specific component-based program;

in a first transaction,

on issuing by a client program running on a first client machine in the distributed computing system a first method invocation to the server-side queued component having passed therein a reference for a client-specific queued component, recording data representative of the first method invocation into a first method invocations message, wherein said recording comprises automatically and transparently to the client program marshaling a reference to the second message queue with the data representative of the first method invocation into the first method invocations message,~~;~~ and

submitting the first method invocations message to the first message queue;

in a second transaction,

retrieving the first method invocations message from the first message queue at the server machine~~;~~,

unmarshaling the data representative of the first method invocation from the first method invocations message~~;~~,

invoking per the first method invocation a method of the server-side queued component, wherein said invoking comprises passing a reference for the client-specific queued component to the server-side queued component~~;~~, and

on invoking by the server-side queued component a method of the client-specific queued component using the reference passed to the server-side queued component, automatically and transparently to the server-side queued component recording data representative of the server-side queued component's method invocations using the reference passed to the server-side queued component into a second method invocations message and submitting the second method invocations message to the second message queue, whereby the server-side queued component's method invocations are queued for the client-specific queued component.

13. (original) In a distributed computing system where a client program, a server queued component, and a client queued component are run on at least one computing machine, a method of transactional queued component interface passing to convey results of processing in the server queued component to the client queued component, the method comprising:

initiating a first transaction for encompassing processing of the client program;

on request of the client program for a reference to the server queued component, creating a first method invocation recorder in the first transaction and returning a reference for invoking methods of the first method invocation recorder to the client program;

on request of the client program for a reference to the client queued component, creating a second method invocation recorder in the first transaction and returning a reference for invoking methods of the second method invocation recorder to the client program;

on a first method invocation of the client program made using the reference for invoking methods of the first method invocation recorder and in which the reference for invoking methods of the second method invocation recorder is passed as a parameter, recording by the first method invocation recorder data of the first method invocation into a first method invocations message and marshaling the second method invocation recorder into the data of the first method invocation;

upon committal of the first transaction, submitting the first method invocations message into a first message queue associated with the server queued component;

creating the server queued component for processing any method invocations recorded in the first method invocations message in a second transaction;

retrieving the first method invocations message from the first message queue within the second transaction;

unmarshaling the second method invocation recorder from the data of the first method invocation in the retrieved first method invocation message such that the second method invocation recorder is re-created in the second transaction;

invoking a method of the server queued component in the second transaction per the first method invocation, and passing a reference for invoking methods of the re-created second method invocation recorder to the server queued component;

on a second method invocation made by the server queued component using the reference for invoking methods of the re-created second method invocation recorder, recording by the second method invocation recorder data of the second method invocation into a second method invocations message;

upon committal of the second transaction, submitting the second method invocations message into a second message queue associated with the client queued component;

creating the client queued component for processing any method invocations recorded in the second method invocations message in a third transaction; and

invoking a method of the client queued component in the third transaction per the second method invocation;

whereby results of processing the first method invocation by the server queued component are conveyed to the client queued component.

14. (currently amended) A computer-readable medium having thereon a computer-executable method invocation queuing system program comprising:

a queued component recorder constructor operating on request of a client program to obtain a queued component reference to create a method invocation recording component and return a reference for such method invocation recording component to the client program;

in a first transaction,

a first method invocation recording component created by the queued component recorder constructor responsive to a first request of a client program to obtain a first reference for a first queued component~~;~~, and

a second method invocation recording component created by the queued component recorder constructor responsive to a second request of the client program to obtain a second reference for a second queued component, the second queued component having a reference passing method accepting a passed object reference as a parameter thereto, the second method invocation recording component operating in response to an invocation of the reference passing method made on the second method invocation recording component having a reference for the first method invocation recording component passed as the parameter to marshal the first method invocation recording component into a data stream representative of the invocation into a message for queuing into a message queue associated with the second queued component; and

in a second transaction,

a queued method invocations playing component operating to retrieve the message from the message queue, the queued method invocations playing component further operating in response to the message to unmarshal the data stream, to re-create the first method invocation recording component, and to invoke the method on the second queued component with a reference for the re-created method invocation recording component passed as the parameter.

15. (canceled)

16. (currently amended) A computer-readable medium having thereon a computer-executable method invocation queuing system program comprising:

a queued method invocations playing component operating to retrieve a method invocations message from a message queue associated with a first queued component, the first queued component having a reference passing method accepting a passed object reference as a parameter thereto during a first transaction, the queued method invocations playing component further operating in response to a message containing a data stream representative of an invocation of the reference passing method having a reference for a method invocation recording component of a second queued component passed as the parameter to unmarshal the data stream, to re-create the method invocation recording component during a second transaction, and to invoke the method on the first queued component with a reference for the re-created method invocation recording component passed as the parameter;

wherein the reference to the re-created method invocation recording component is passed to the first queued component so the first queued component can record a result to the second queued component.

17. (previously presented) The computer-readable medium of claim 16 wherein the result is a method invocation intended for the second queued component.

18. (previously presented) The computer-readable medium of claim 16 wherein the result is a result of the method invoked on the first queued component.